In-Class Midterm

(2:35 PM – 3:50 PM : 75 Minutes)

- This exam will account for either 15% or 30% of your overall grade depending on your relative performance in the midterm and the final. The higher of the two scores (midterm and final) will be worth 30% of your grade, and the lower one 15%.
- There are four (4) questions, worth 75 points in total. Please answer all of them in the spaces provided.
- There are 16 pages including four (4) blank pages and two (2) pages of appendices. Please use the blank pages if you need additional space for your answers.
- The exam is *open slides.*

Good Luck!

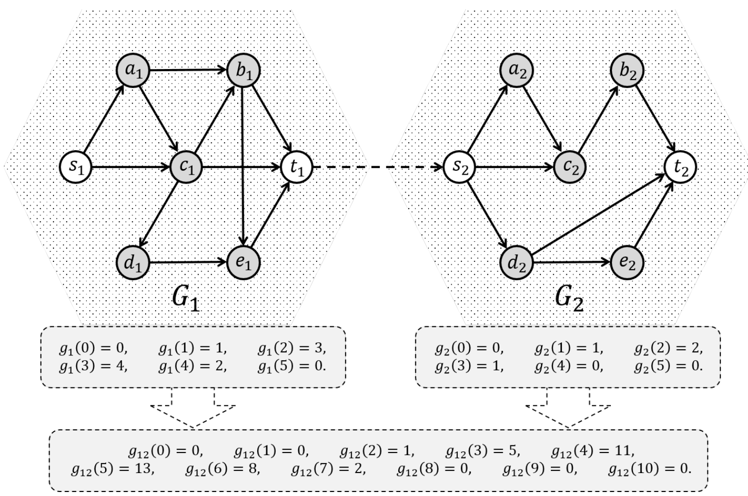
Question	Pages	Score	Maximum
1. Counting Paths	2–4		20
2. A Schönhage-Strassen-like Recurrence	6–8		25
3. Closest Pair of Points	10–11		20
4. An Impossible Priority Queue	13		10
Total			75

Name: ______________________________

Question 1. [**20 Points**] **Counting Paths.** Suppose you are given two directed graphs[1] G_1 and G_2 containing $n+2$ nodes each for some $n \geq 0$. For $i \in \{1, 2\}$, G_i includes two special nodes — a *source node* s_i with no incoming edges[2] and a *target node* t_i with no outgoing edges[3]. These two nodes are called *external nodes* while the rest are called *internal nodes*. The figure below shows an example with $n = 5$ in which the internal nodes are colored grey and the external nodes are white. Let $g_i(k)$ denote the number of paths in G_i that go from s_i to t_i and pass through exactly k internal (i.e., grey) nodes. For example, in the figure below $g_1(3) = 4$ which represents the following 4 paths:

$$\begin{aligned} s_1 \to a_1 \to b_1 \to e_1 \to t_1,\\ s_1 \to a_1 \to c_1 \to b_1 \to t_1,\\ s_1 \to c_1 \to b_1 \to e_1 \to t_1\\ \text{and } s_1 \to c_1 \to d_1 \to e_1 \to t_1. \end{aligned}$$

Suppose for $0 \leq k \leq n$, all $g_1(k)$ and $g_2(k)$ values are known to you.



Now suppose you connect G_1 and G_2 by putting an edge directed from t_1 to s_2. For $0 \leq k \leq 2n$, let $g_{12}(k)$ denote the number of paths from s_1 to t_2 that pass through exactly k internal (i.e., grey) nodes. The figure above shows an example in which $g_{12}(3) = 5$ representing the following 5 paths:

$$\begin{aligned} (s_1 \to c_1 \to t_1) \to (s_2 \to c_2 \to b_2 \to t_2),\\ (s_1 \to c_1 \to t_1) \to (s_2 \to d_2 \to e_2 \to t_2),\\ (s_1 \to a_1 \to b_1 \to t_1) \to (s_2 \to d_2 \to t_2),\\ (s_1 \to a_1 \to c_1 \to t_1) \to (s_2 \to d_2 \to t_2)\\ \text{and } (s_1 \to c_1 \to b_1 \to t_1) \to (s_2 \to d_2 \to t_2). \end{aligned}$$

[1] e.g., road networks with one-way roads
[2] e.g., incoming roads
[3] e.g., outgoing roads

1(a) [**5 Points**] For any given integer $k \in [0, 2n]$, show that $g_{12}(k)$ can be computed from g_1's and g_2's in $\mathcal{O}(n)$ time.

1(b) [**15 Points**] Show that for $0 \leq k \leq 2n$, one can compute all $g_{12}(k)$ values simultaneously in $\mathcal{O}(n \log n)$ time.

Use this page if you need additional space for your answers.

Question 2. [25 Points] A Schönhage-Strassen-like Recurrence. Consider the following recurrence (for $n \geq 2$) which is similar to the recurrence that arises during the analysis of the *Schönhage-Strassen algorithm* for multiplying large integers.

$$T(n) = \begin{cases} \Theta(1) & \text{if } 2 \leq n \leq 8, \\ n^{\frac{2}{3}} T\left(n^{\frac{1}{3}}\right) + n^{\frac{1}{3}} T\left(n^{\frac{2}{3}}\right) + \Theta(n \log n) & \text{otherwise.} \end{cases}$$

2(a) [**4 Points**] Show that the recurrence above can be rewritten as follows, where $T(n) = nS(n)$.

$$S(n) = \begin{cases} \Theta(1) & \text{if } 2 \leq n \leq 8, \\ S\left(n^{\frac{1}{3}}\right) + S\left(n^{\frac{2}{3}}\right) + \Theta(\log n) & \text{otherwise.} \end{cases}$$

2(b) [**4 Points**] Show that the recurrence in 2(a) can be rewritten as follows, where $P(x) = S(2^x)$.

$$P(x) = \begin{cases} \Theta(1) & \text{if } 1 \leq x \leq 3, \\ P\left(\frac{x}{3}\right) + P\left(\frac{2x}{3}\right) + \Theta(x) & \text{otherwise.} \end{cases}$$

2(c) [**9 Points**] Solve the recurrence from part 2(b) to show that $P(x) = \Theta(x \log x)$.

2(d) [**8 Points**] Use your results from part 2(c) to show that $T(n) = \Theta(n \log n \log\log n)$.

Use this page if you need additional space for your answers.

Question 3. [20 Points] Closest Pair of Points. Consider the algorithm Closest-Pair given below that finds the closest pair of points among a given set of points in the plane.

Closest-Pair(P, n)

Input: A set $P = \{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)\}$ of n points in the plane. Assume for simplicity that (a) $n = 2^k$ for some integer $k > 0$, (b) all x_i's are distinct, and (c) all y_i's are distinct.

Output: Two distinct points $p_i, p_j \in P$ such that the distance between p_i and p_j is the smallest among all pairs of points in P.

Algorithm:

1. ***if*** $n = 2$ ***then return*** $\langle p_1, p_2 \rangle$
2. ***else***
3. Find a value x such that exactly $\frac{n}{2}$ points in P have $x_i < x$, and the other $\frac{n}{2}$ points have $x_i > x$
4. Let L be the subset of P containing all points with $x_i < x$
5. Let R be the subset of P containing all points with $x_i > x$
6. $\langle p_L, q_L \rangle \leftarrow$ Closest-Pair(L, $\frac{n}{2}$)
7. $\langle p_R, q_R \rangle \leftarrow$ Closest-Pair(R, $\frac{n}{2}$)
8. $d_L \leftarrow$ distance between p_L and q_L
9. $d_R \leftarrow$ distance between p_R and q_R
10. $d \leftarrow \min\{\, d_L,\ d_R \,\}$
11. Scan P and remove each $p_i = (x_i, y_i) \in P$ with $x_i < x - d$ or $x_i > x + d$
12. Sort the remaining points of P in increasing order of y-coordinates
13. Scan the sorted list, and for each point compute its distance to the 7 subsequent points in the list. Let $\langle p_M, q_M \rangle$ be the closest pair of points found in this way.
14. Let $\langle p, q \rangle$ be the closest pair among $\langle p_L, q_L \rangle$, $\langle p_R, q_R \rangle$ and $\langle p_M, q_M \rangle$
15. ***return*** $\langle p, q \rangle$

3(a) [**10 Points**] Argue that for a set of n points, steps 3–5 take $\mathcal{O}(n)$ time while steps 8–15 take $\mathcal{O}(n \log n)$ time.

3(b) [**10 Points**] Let $T(n)$ be the running time of CLOSEST-PAIR on a set of n points. Write a recurrence relation for $T(n)$ and solve it.

Use this page if you need additional space for your answers.

Question 4. [**10 Points**] **An Impossible Priority Queue.** Consider a (comparison-based) priority queue Q (for real numbers) that supports the following operations.

Make-Queue(Q)**:** Create an empty queue Q.

Insert(Q, x)**:** Insert item x into Q.

Increase-Key(Q, x, k)**:** Increase the key of item x to k assuming $k \geq$ current key of x.

Find-Min(Q)**:** Return a pointer to an item in Q containing the smallest key.

Delete-Min(Q)**:** Delete an item with the smallest key from Q and return a pointer to it.

4(a) [**10 Points**] Suppose Q supports Insert and Increase-Key operations in $\mathcal{O}(1)$ amortized time each, and Delete-Min operations in $\mathcal{O}(\log n)$ worst-case time each, where n is the number of items in Q. It also supports the Make-Queue operation and every Find-Min operation in $\mathcal{O}(1)$ worst-case time.

Argue that such a priority queue cannot exist.

Use this page if you need additional space for your answers.

Appendix: Recurrences

Master Theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise,} \end{cases}$$

where, $\frac{n}{b}$ is interpreted to mean either $\left\lfloor \frac{n}{b} \right\rfloor$ or $\left\lceil \frac{n}{b} \right\rceil$. Then $T(n)$ has the following bounds:

Case 1: If $f(n) = \mathcal{O}\left(n^{\log_b a - \epsilon}\right)$ for some constant $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.

Case 2: If $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ for some constant $k \geq 0$, then $T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$.

Case 3: If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some constant $\epsilon > 0$, and $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Akra-Bazzi Recurrences. Consider the following recurrence:

$$T(x) = \begin{cases} \Theta(1), & \text{if } 1 \leq x \leq x_0, \\ \sum_{i=1}^{k} a_i T(b_i x) + g(x), & \text{otherwise,} \end{cases}$$

where,

1. $k \geq 1$ is an integer constant,
2. $a_i > 0$ is a constant for $1 \leq i \leq k$,
3. $b_i \in (0, 1)$ is a constant for $1 \leq i \leq k$,
4. $x \geq 1$ is a real number,
5. x_0 is a constant and $\geq \max\left\{\frac{1}{b_i}, \frac{1}{1-b_i}\right\}$ for $1 \leq i \leq k$, and
6. $g(x)$ is a nonnegative function that satisfies a polynomial growth condition (e.g., $g(x) = x^\alpha \log^\beta x$ satisfies the polynomial growth condition for any constants $\alpha, \beta \in \Re$).

Let p be the unique real number for which $\sum_{i=1}^{k} a_i b_i^p = 1$. Then

$$T(x) = \Theta\left(x^p\left(1 + \int_1^x \frac{g(u)}{u^{p+1}} du\right)\right).$$

Appendix: Computing Products

Integer Multiplication. Karatsuba's algorithm can multiply two n-bit integers in $\Theta\left(n^{\log_2 3}\right) = \mathcal{O}\left(n^{1.6}\right)$ time (improving over the standard $\Theta\left(n^2\right)$ time algorithm).

Matrix Multiplication. Strassen's algorithm can multiply two $n \times n$ matrices in $\Theta\left(n^{\log_2 7}\right) = \mathcal{O}\left(n^{2.81}\right)$ time (improving over the standard $\Theta\left(n^3\right)$ time algorithm).

Polynomial Multiplication. One can multiply two n-degree polynomials in $\Theta\left(n \log n\right)$ time using the FFT (Fast Fourier Transform) algorithm (improving over the standard $\Theta\left(n^2\right)$ time algorithm).